AD-A045 900 MASSACHUSETTS UNIV AMHERST F/G 9/2
DEVELOPMENT OF QUALITY LANGUAGES.(U)
OCT 77 H F LEDGARD DAHC04-74-G-0139
UNCLASSIFIED ARO-12246.5-M NL

1 OF 1
ADA
045900

END
DATE FILMED
11-77
DDC

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARO 12246.5-M

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 12246.5-M	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Development of Quality Languages,	5. TYPE OF REPORT & PERIOD COVERED: Final Report. 1 Jun 74 - 15 Oct 77
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Henry F. Ledgard	8. CONTRACT OR GRANT NUMBER(s): DAHC04-74-G-0139; 75 G 0173 DAAG29-76-G-0216
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Massachusetts, Amherst, Massachusetts 01003	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 12 11p.
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709	12. REPORT DATE: Oct 77
	13. NUMBER OF PAGES: 9
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): DAHC04-74-G-0139, DAHC04-75-G-0173	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Language design
Quality software
Formal definition
Human engineering
Computer programs
Programming languages

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Over the past decade the computer software area has faced severe problems over the quality of software. Resulting systems have often been plagued with poor performance, errors, and frequent expensive modifications. The research reported here was directed at one key element in the production of software - the programming language. Three somewhat overlapping approaches were explored: (1) the development of desired properties in languages, (2) the development of new definition approaches and the clarification of existing approaches, and (3) the emerging importance of the role of human factors in language design.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 220 200
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Final Report

on

DEVELOPMENT OF QUALITY LANGUAGES

Submitted To

U.S. Army Research Office
Durham, NJ

Principal Investigator:
Henry F. Ledgard

Duration:
June 1, 1974 to October 15, 1977

Contract Numbers:
DAHC04-74-G-0139
DAHC04-75-G-0173
DAAG29-76-G-0216

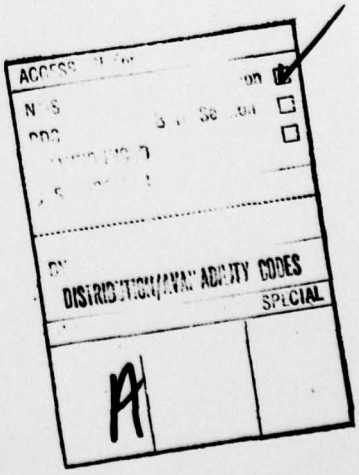
Keywords: Language Design, Quality Software, Formal Definition, Human Engineering

ACKNOWLEDGMENT

I would like to express my appreciation to the Army Research Office in supporting one of the most rewarding efforts I have undertaken.

During the three years of grant support, I have been able to work on some of the most critical problems in language design; to integrate this work with the software quality group at Fort Monmouth; to serve as a design consultant on the common language effort; and to develop a recent concern for one of the most critical factors in language design--that of human engineering.

All of these items were stimulated by this research contract, and the personal support by the Army Research Office has been a major factor in making the past three years a worthwhile period for me.



A. PROBLEMS STUDIED

"We propose here to investigate the underlying structures in the design of algorithmic programming languages, to integrate and expand on the recent developments in software quality and formal definition, and thereby to develop a set of new linguistic structures for programming languages."

Quote taken from the original proposal
June, 1974

B. PEOPLE SUPPORTED: Major

Michael Marcotty, Frederic Richard, Andrew Singer

PEOPLE SUPPORTED: Minor

Randy Chow, Jon Hueras, Joseph Kasprzyk, Gary Madelung

C. LIST OF PUBLICATIONS: Final

1) Henry F. Ledgard
Production Systems: A Notation for Defining Syntax and Translation
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Volume SE-3(2), March 1977, p. 105-124

2) Michael Marcotty, Henry Ledgard, and Gregor Bochmann
A Sampler of Formal Definitions
COMPUTING SURVEYS, Volume 8(2), June 1976, p. 191-276

3) Henry Ledgard and Michael Marcotty
A Genealogy of Control Structures
COMMUNICATIONS OF THE ACM, Volume 19(11), November 1976, p. 601-608

4) Henry Ledgard and Robert Taylor
The Two Views of Data Abstraction (a special introduction)
COMMUNICATIONS OF THE ACM, Volume 20(6), June 1977, p. 382-384

5) Henry Ledgard and Andrew Singer
Formal Definition and Design
Presented at the 1977 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD

6) Frederic Richard and Henry F. Ledgard
A Reminder for Language Designers
To appear in SIGPLAN Notices, Winter 1977-78

7) Henry Ledgard and Andrew Singer
The Case for Research in Human Engineering
To appear in the Department of Defense Handbook on Research Directions, September 1977
(revised version submitted to Communications of the ACM)

LIST OF PUBLICATIONS: In Progress at the end of Granting Period

8) Andrew Singer
Ph.D. Thesis, "Human Factors and Formal Definition"
(title has not been finalized)

This work shows the importance of formal definition during the design process.

9) Henry Ledgard and Andrew Singer
Formal Definition and Design
(to be submitted to Computing Surveys)

This is an attempt to make a case for a substantial redirection of formal definition techniques. The basic case to be made is that formal definitions should serve in the design of software much as an architectural drawing serves in the design of a physical structure.

10) Andrew Singer, Henry Ledgard, and Jon Hueras
Software Design and the Human Interface

The work is an attempt to demonstrate the kind of human engineering that can be done for software users.

D. CONFERENCES HELD:

ACM Conference on Data Abstraction, Salt Lake City, UT, March 1976

1. INTRODUCTION

Over the past decade the computer software area has faced severe problems over the quality of software. Resulting systems have often been plagued with poor performance, errors, and frequent expensive modifications. The research reported here was directed at one key element in the production of software: the programming language.

During the granting period, three somewhat overlapping approaches were explored.

1) the development of desired properties in languages,

2) the development of new definition approaches and the clarification of existing approaches,

3) the emerging importance of the role of human factors in language design.

2. DESIRED PROPERTIES FOR HIGHER LEVEL LANGUAGES:

2.1 Control Structures: The first major accomplishment in this area was on control structures. In past years there have been perhaps hundreds of efforts treating this important issue. While it may be argued that the control structure issue has been entirely over worked, the importance of definitive works in this area remains. Our basic approach to this problem was to analyze all existing work, notably the theoretical work of Kosaraju, in an attempt to distill a recommendation for a solution to the control structure problem.

The basic result of our study was a deep-rooted and far-reaching committment to the use of 1-in, 1-out control structures.

One major realization during this period was that, from the programmer's point of view, theoretical results based on the conversion of one program form to another under restricted conditions may not be of practical significance. After numerous studies of existing work and many, many examples, we found the case for higher order control structures unconfirmed. We have concluded from both theoretical and practical work that 1-in, 1-out control structures provide a solid basis for language design. They provide tools for proper flow of control, and furthermore, force the programmer to think ahead in devising clean solutions. This work was one of the cornerstones of our research and is described in [3].

2.2 Language Design Guidelines: On another front, a serious attempt was made to distill a number of recommendations for the design of languages. It is all too easy to make statements that are not carefully supported. As a result of our general experience in the language area, we put together a set of guidelines for language designers. These guidelines treat issues like the overall complexity of a language, the design of function and procedure facilities, the importance of program layout, the redundancy of information and internal documentation. This work is described in [6].

2.3 Asychronous Control Structures: During the course of the research, two problems were studied and yielded no significant results. The first was an investigation of asychronous control

structures. It was our major assumption that the relaxation of strict sequential flow of control in a conventional programming language would considerably reduce some of the sources of complexity in programming.

In this study we considered the separation between data flow and the communication between modules via input-output buffers. A small control language was defined for the description of the interprocesses activities. The result of this investigation was that the asynchronous approach was more natural only for certain classes of programs (such as simulation). In general, we had to put much of our programming effort into organizing data flow in order to get a good asynchronous solution. We also found that, given a set of proposed program modifications, the asynchronous solution was no better than the conventional top-down solution. Our major conclusion is that the asynchronous approach has been overrated, and in their present form are not generally suitable for conventional languages.

2.4 A Measure of Syntactic Clarity: Another effort which did not bear fruit was an investigation of syntactic clarity. Using a variation of the LR(k) property for grammars, an attempt was made to define a formal measure of syntactic clarity. An abstract algorithm for computing this measure was devised. Using the algorithm, experimental tests of the measure were carried out by hand on two "mini-languages" and their variations.

While the results seemed promising, the algorithm did not seem to be completely defined. The amount of effort necessary to pursue the research further and the inherent difficulty in getting a successful result motivated discontinuation of the work.

2.5 Data Abstraction: During the grant period a conference on data abstraction was organized by myself and was held in Salt Lake City. A paper resulting from this conference was part of a sp al issue of the Communications of the ACM devoted to the rence [4]. From an organizational point of view, the conference was a splendid success. It brought together key people in the data abstraction area and led to a thoughtful and pleasant exchange of ideas. From a research point of view, the conference exposed several problems with this area.

The computer sciences have seen many new developments with exciting promise, followed by a slow follow-up as the details of the problems result in ever increasing difficulties. This has certainly been the case in the data abstraction area. Progress has been slow, and it has come in small doses.

3. FORMAL DEFINITION

In the area of formal definition we feel that we have had some great successes. Our major activity was on the use of formal definition in the language design process.

3.1 The Sampler: This effort has resulted in a major paper, "A Sampler of Formal Definition," which appeared in Computer

Surveys [2]. This paper compares the use of VDL, Knuth's Attribute Grammars, van Wijngaarden's W-Grammars, Hoare's Axiomatic Approach, and my own Production System notation

This paper attempted to set a new standard for completeness and readability in formal definitions. In addition, the paper exposed the weaknesses of four major methods of formal definition. Over the years to come we feel that the significance of this paper will continue to grow.

3.2 Production Systems: One of the major formal definitions in the above mentioned paper has been use of my own "production Systems," a new notation for defining computer languages. A paper describing Production Systems was completed [1]. Judging from the work on the Sampler, Production Systems may have a bright future in the development of quality languages.

3.3 The Role of Formal Definition: All of this effort yielded a deepening insight into the proper uses of formal definition. We have come to believe that the major benefit for formal definition is as a basis for detailed design of computer languages.

In a sense, the writing of formal definitions as a design tool is programming at the very highest level. It provides a view of the language from which design can proceed at a much more rapid rate and most importantly, with a much higher quality.

Our case for this issue has not been fully prepared and will remain as unfinished business during the granting period. We expect to resolve this situation by publishing a paper on our views of formal definition in the very near future [9].

5. HUMAN FACTORS IN LANGUAGE DESIGN

During the closing months of the granting period, a new and deep-rooted committment was made to the area of human factors in the language design process. The recent surge of research in the area of software quality has concentrated on programming technologies, control structures, management of large systems, and reliability. All of these issues are aimed at producing better quality programs; yet, none of them addresses directly the problem of producing programs that are easier to use. Surely, no one would argue that the ease of use is not a factor in the "quality" of software.

In our work in human factors, two papers were completed. The first entitled "The Case for Research in Human Engineering" [7] is to appear in the Department of Defense Handbook on Research Directions. This paper describes the problems in existing systems and points out that there is no organized body of human factors knowledge to guide language designers. The basic contention of this paper is that a much greater priority must be given to research in human engineering.

On another front, we have attempted to demonstrate by example the kind of beautiful human engineering that can be done by software engineers [10]. This work is still in progress and will remain as unfinished business at the end of the granting period.

In any case, we are deeply committed to this area and feel that the long range payoffs will be enormous.